Development report

展示

- ppt连接：
- 【金山文档 | WPS云文档】 演示ppt第一版
  https://kdocs.cn/l/cc11JAzLurit

运行须知

爬虫与部署了各类模型的云服务器位于学院服务器，数据库部署在云端。现已停止维护，存在无法连接的可能。

爬虫

注意事项

- 开发和测试使用python3.11，版本不要太低应该都可以

  不推荐python3.12，后续安装paddleocr时会报错缺少lmdb，需要自己配置
- 日志记录功能目前在Linux环境可用，Windows下会报错（可以禁用日志功能）

运行前准备

- 请根据setting要求配置爬虫cookie、数据库与情感分析云服务token
- 安装百度飞桨paddlepaddle（用于运行ocr）

  参考官方安装文档

  （以2.6版本，Linux系统，cpu驱动，conda安装为例）

  ```
  conda install paddlepaddle==2.6.1 --channel
  https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/Paddle/
  ```
- 安装依赖

  终端进入项目路径`SchoolMarket`后，输入以下命令

  ```
  pip install -r requirements.txt
  ```

requirements_conda.txt是conda list -e生成

requirements_env.txt是pip list --format=freeze生成

爬虫运行环境可以参考以上两个txt文件

运行和结束

- ssh退出后，Linux服务器上依然运行：`nohup <command> &`
  (例如`nohup python main.py &`)
  - 错误重定向输出至error.log：`nohup python main.py > logs/error.log 2>&1 &`
  - 不记录终端输出（日志中已记录）：`nohup python main.py > /dev/null 2>&1 &`
- 查看进程：`ps -ef | grep main.py`*注意父子进程*
  - 父进程是运行main.py
  - 子进程运行爬虫，每次运行结束关闭该进程
- 结束：`kill -9 <process_id>`

数据库

远程连接：

host="sse-21311572-train.mysql.rds.aliyuncs.com",

port="3307",

user="SSE_user1",

password="SSE_user1test",

database="sse_training",

后端

安装依赖

```
pip install -r requirements.txt
```

启动项目

```
python3 manage.py runserver
```

前端

在`hotpoints`目录下：

- 安装依赖

  ```
  npm install
  ```

- 启动

  ```
  npm run serve
  ```

大模型

1. 大模型参数和运行配置

- **Baichuan2-13B-4bits**：
  - 参数量为13B的模型
  - 并进行了4bits的量化，提升了模型的运行效率和内存使用率，使得模型可以在 **20GB 显存**的限制下运行。
  - 加载基于IEPile训练的 LoRA 权重，针对性地对**事件提取**等任务进行专门优化，并且有效地**稳定输出格式**，便于后端处理。
- **推荐配置**：
  - NVIDIA RTX3090/4090 及以上
  - 显存大于 20 GB

2. 运行

在`Baichuan2`目录下：

- 安装依赖

```
pip install -r requirements.txt
```

- 启动 ngrok 内网穿透

```
ngrok http --domain=akita-famous-sincerely.ngrok-free.app 8000
```

- 启动 Baichuan2 大模型 API

```
python OpenAI_api.py
```

代码结构

主要保留二次开发的可能，说明开发需要用到的某某文件用处用途，写完后我会统一放在子文件的readme里

爬虫

```
SchoolMarket/
├── config                  // 自定义的配置文件
|   ├── db_config.py              // 数据库连接配置
|   └── spider_config.py          // 爬虫配置
├── __init__.py
├── items.py                // scrapy的item（暂存爬取数据）
├── logs                    // 日志以及日志备份（7天）
|   ├── scrapy_log.log
|   ...
|   └── scrapy_log.log.2024-07-13
├── main.py                 // 运行文件
├── middlewares.py          // scrapy中间件
├── pipelines.py            // scrapy流水线（处理爬取的数据）
├── settings.py             // scrapy设置
├── spiders                 // scrapy爬虫
|   ├── __init__.py
|   ├── home.py                   // 首页爬虫
|   ├── hot.py                    // 热榜爬虫
|   └── trace.py                  // 追踪爬虫
└── utils                   // 自定义组件
    ├── __init__.py
    ├── calculate_hot.py            // 计算热度值和热度增长
    ├── connect_pool.py             // 数据库连接池
    ├── correlation_cal_bert.py     // 计算帖子相关度（用bert模型）
    ├── correlation_cal_w2c.py      // 计算帖子相关度（用w2c）
    ├── database.py                 // 数据库操作
    ├── ocr.py                      // 光学字符识别
    ├── sentiment_Rateing.py        // 情感分析
    └── stopwords         // 各种停用词表（情感分析分词用）
```

```
        ├── stopwords_all.txt
        ├── .....
        └── stopwords_scu.txt
```

数据库

相关维护更新逻辑已接入爬虫子项目

后端

```
├── Network_Hotspots_Mining          # 项目根目录，项目名为Network Hotspots Mining
│   ├── app                          # Django应用目录
│   │   ├── admin.py                     # Django admin后台配置
│   │   ├── apps.py                      # Django应用定义文件
│   │   ├── controller                   # 控制器模块，存放处理业务逻辑的脚本
│   │   │   ├── LLM.py                       # 处理语言模型相关功能的脚本
│   │   │   ├── __pycache__                  # Python编译后的字节码缓存目录
│   │   │   │   ├── LLM.cpython-310.pyc          # LLM.py对应的字节码文件
│   │   │   │   └── single_pass.cpython-310.pyc      # single_pass.py对应的字节码文件
│   │   │   └── single_pass.py               # single-pass聚类算法文件
│   │   ├── data                         # 存放数据文件的目录
│   │   │   ├── data1.txt                    # 示例数据文件1
│   │   │   └── stop_words.txt               # 停用词文件，常用于文本预处理
│   │   ├── __init__.py                  # 初始化文件，用于定义模块属性
│   │   ├── migrations                   # 数据库迁移文件目录，记录模型变更历史
│   │   │   ├── 0001_initial.py              # 初始数据库迁移脚本
│   │   │   ├── __init__.py                  # 迁移模块初始化文件
│   │   │   └── __pycache__                  # 迁移脚本的字节码缓存
│   │   ├── misc                         # 杂项或辅助脚本目录
│   │   │   ├── clear_db.py                  # 清空数据库脚本
│   │   │   ├── data_processing.py           # 数据处理脚本
│   │   │   ├── __pycache__                  # 杂项脚本的字节码缓存
│   │   │   └── test.py                      # 测试脚本或示例脚本
│   │   ├── models.py                    # 定义Django模型（数据库表结构）的文件
│   │   ├── __pycache__                  # app模块中各py文件的字节码缓存
│   │   ├── result                       # 存放处理结果或报告的目录
│   │   │   ├── ...                          # 各种结果文件
│   │   ├── tasks.py                     # Celery异步任务定义文件
│   │   ├── tests.py                     # 单元测试文件
│   │   ├── util                         # 工具函数或模块目录
│   │   │   ├── data_analysis.py             # 数据分析工具脚本
│   │   │   ├── __pycache__                  # 工具模块的字节码缓存
│   │   │   └── util.py                      # 工具函数主文件
│   │   └── views.py                     # 视图函数定义，处理HTTP请求和响应
│   ├── manage.py                    # Django项目管理命令入口
│   └── Network_Hotspots_Mining      # 与项目同名的目录，可能包含了项目级别的配置
│       ├── asgi.py                      # ASGI服务器配置文件，用于部署Web应用
│       ├── celery.py                    # Celery配置文件，用于任务队列设置
│       ├── __init__.py                  # 项目级别初始化文件
│       ├── __pycache__                  # 项目级别配置文件的字节码缓存
│       ├── settings.py                  # Django项目设置文件
│       ├── urls.py                      # URL路由配置文件
│       └── wsgi.py                      # WSGI服务器入口文件，用于生产环境部署
├── README.md                        # 项目说明文档，包含安装、使用等指导信息
```

```
└── requirements.txt                # 项目依赖列表文件，用于pip安装所需第三方库
```

前端

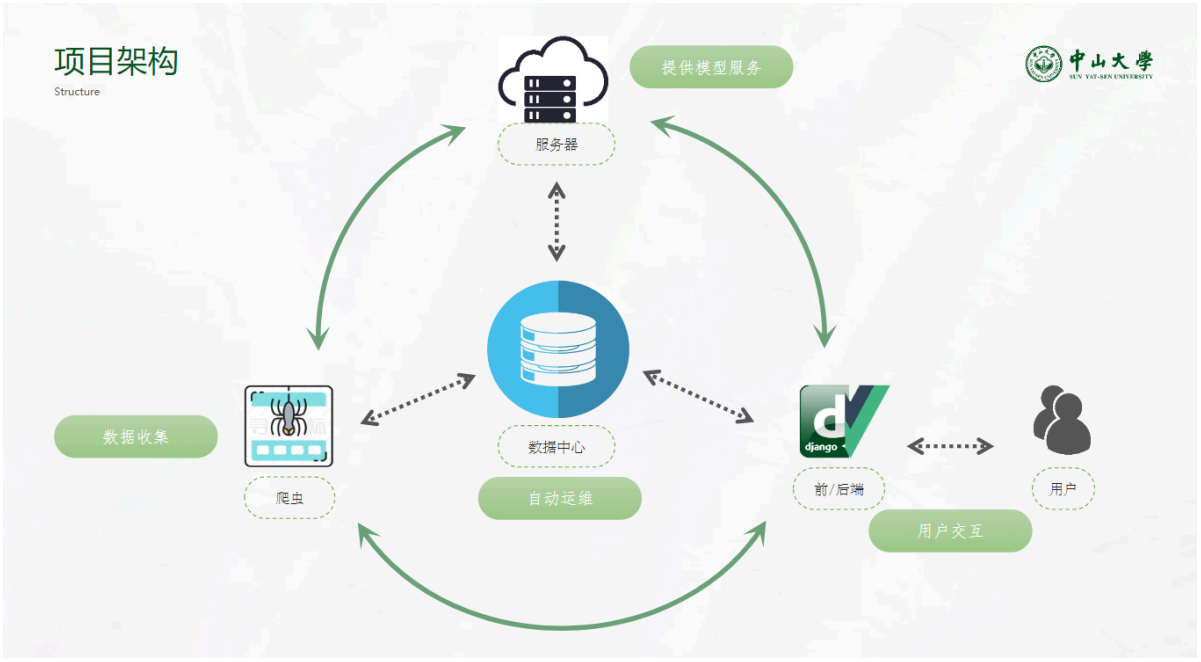
```
src/
|
├── api/                        // 网络请求
|   ├── xxx.js
|   └── xxx.js
|
├── assets/                     // 资源文件
|   └── xxx
|
├── components/                 // 组件
|   ├── xxx.vue/
|   ...
|   └── xxx.vue2/
|
├── router/                     // 路由
|   └── index.js
|
├── utils/                      // 自定义功能函数
|   └── xxx
|
├── views/                      // 页面文件
|   ├── HomeView/
|   └── Topic/
|
└── App.vue                     // 入口文件
```

大模型

```
Baichuan2/
|
├── baichuan-inc/                       // LoRA 权重
|   ├── Baichuan2-7B-Chat-4bits
|
└── OpenAI_api.py                       // 大模型 API 文件
```

技术报告

整体架构



开发工具

- 使用github进行代码协作开发（暂未开源）
- 使用apifox进行接口开发管理

张隽滔

技术工作

- 爬虫

  基于Scrapy框架开发，定时对“赞噢校园集市·中大站”的帖子内容及其评论进行爬取。根据热度值以及热度值的增长率，追踪可能的热点帖子。每天的爬虫信息会存储到相应日期的.log文件中，会定时清理过旧的日志。

- OCR

  采用百度飞桨(paddle)的开源轻量化模型ppocr_v4，对校园集市帖子中的图片进行OCR，实现图转文，为帖子补充信息

- 数据库连接和操作

  维护数据库连接池，避免频繁连接数据库导致性能下降。参数化执行SQL语句，防止注入攻击和提高性能。

- 设计爬虫项目框架逻辑

  将OCR、情感分析、热度计算、相关度计算、数据库连接等功能封装，在爬虫的相应位置调用，实现爬虫数据预处理以及存入Mysql数据库。

遇到的问题以及解决办法

- 赞噢校园集市页面只能在微信浏览器打开
  - 问题：

    微信浏览器不允许打开开发者工具进行页面调试，也导致难以分析页面元素进行过滤。若用一些个人开发的插件魔改微信浏览器，有封号的风险。

  - 解决方法：

用抓包工具Charles，分析打开集市不同页面的数据流。发现传输集市内容数据的json文件以及相关图片，可以通过一般的网络请求获取。只需要使用合法的用户token，改写url和相应的请求参数，即可获取想要的页面信息。

苏东鹏（组长）

调研阶段

- 负责项目经理工作，组织小组项目调研、开发。完成需求分析、产品设计，绘制产品原型图
- 组织前期技术调研、项目分析，制作本项目全阶段解决方案/技术路线
- 与张共同探索校园集市爬虫框架设计、确立数据中心结构及维护框架
- 负责数据中心热度算法、情感分析、数据库维护、舆论评级算法等的调研

开发阶段

- 以张的爬虫端为基础，建立云端数据库，完成数据库维护、爬虫追踪、热度算法等数据库的逻辑设计与代码开发
- 基于百度ERNIE-UIE，完成数据中心的情感分析算法设计与开发
- 基于MacBert等模型，完成数据中心的相关性算法设计与开发
- 基于数据中心，完成舆论评级算法的设计与开发
- 负责大部分后端接口开发

部分技术示例

叶文熙

调研阶段

调研事件总结、事件抽取方法，产出调研知识文档，提出尝试deepKE进行事件抽取的建议。

针对网络热点文本聚类方法和对Python语言下的后端框架进行调研，对实现难度和性能进行评估。

开发阶段

个人从零搭建项目后端部分，完成后端部分开发。

- 实现后端与数据库对接，使后端能从数据库读取数据
- 基于SIngle-Pass算法实现新闻热点聚类
    - 设置停用词列表，形式为文件，通过读取文件可获取停用词列表。
    - 切割句子cut_sentences： 接受文本或文本文件路径作为输入，使用jieba进行分词处理，并将分词后的文本转换为“空格+词”的字符串形式，同时构建文本ID到文本内容的映射。
    - 获取TF-IDF方法 get_tfidf： 将分词后的文本转换为TF-IDF矩阵表示，并将其转换为稠密列表形式返回。
    - 余弦相似度计算方法 cosion_simi： 计算给定向量与所有簇中心向量之间的余弦相似度，并返回最大相似度值及其对应的簇索引。
    - 单遍聚类方法 single_pass： 对输入文本执行single_pass聚类算法。首先调用cut_sentences和get_tfidf方法预处理文本，然后遍历每个文本的TF-IDF向量，根据与已有簇中心的相似度决定是否加入现有簇或创建新簇。最后，将聚类结果保存到指定的JSON文件中。
- 与大模型API对接
    - 使用线程池并行技术对调用大模型API函数进行性能优化
    - 调用LLM API对事件聚类进行总结的同时计算每个聚类的热度和热度上升率
- 接口开发
    - 获取热度上升榜get_speedlist
    - 获取热度总值榜get_hotlist。
- 编写测试脚本（misc目录）
    - 数据预处理脚本data_processing.py
    - 清空数据库指定库表脚本clear_db.py
    - 测试部分后端接口功能脚本test.py

黄炜尧

背景调研：任务

相比传统机器学习模型，大模型通过大量的数据预训练，具备了强大的**泛化能力**，可以完成多种传统机器学习模型难以完成的任务，如事件总结、事件关系判断等，并表现出更高的准确性和稳定性。

除此之外，大模型还具有**自动特征提取**的能力。通过大量的数据的训练，大模型能够自动提取和学习数据中的特征，而无需人工设计复杂的特征工程。这极大地简化了开发流程，提升了开发效率。这也非常适合本项目的任务特征。

因此，我们使用大模型完成项目中的3个任务：

- **事件总结**：利用大模型的深度学习能力，自动识别并总结事件的关键内容和重要信息。
- **聚类总结**：利用大模型处理聚类算法的结果，对每一类进行总结。
- **事件关系**：利用大模型理解和分析事件之间的关系，如前提、结果、补充等。这为**知识图谱**的构建提供了必要的信息。

背景调研：开源大模型对比

在选择适合项目的大模型时，我们需要考虑多个因素，包括模型的语言支持、参数量、灵活性以及适应特定任务的能力。由于项目的计算资源有限，为了平衡效率和精度，我们统一选择参数量为7B左右，或者量化后的13B模型进行测试。

- **OneKE**：
  - **开发者/机构**：由蚂蚁集团和浙江大学联合研发。
  - **主要特点**：这是一个专注于知识抽取的大模型框架，适用于处理结构化知识提取任务。
  - **放弃原因**：尽管OneKE在知识抽取方面表现出色，但它在处理复杂的事件关系任务时灵活度不足，不支持自定义事件关系的抽取和分析，无法建立知识图谱。
- **LLaMA2-7B**：
  - **主要特点**：LLaMA2-7B是一款性能优异的通用大模型，参数量为7B，适用于多种语言处理任务。
  - **放弃原因**：虽然LLaMA2-7B在多语言处理上具有较强的能力，但其在中文任务上的表现并不足以满足我们的需求。由于本次项目主要面向中文环境，需要一个对中文有更深入优化的模型。
- **Baichuan2-13B-4bits**：
  - **主要特点**：Baichuan2是一款专门为中文优化的大模型，参数量为13B但经过4bits量化处理，使得模型既保持了高精度也适应了有限的计算资源。
  - **选择理由**：Baichuan2-13B-4bits的优异中文处理能力以及其量化后的高效性使它成为我们项目的理想选择。它能够有效处理事件总结、聚类总结和事件关系这三个核心任务，同时满足项目对精度和效率的双重需求。
  - 下面是 Baichuan 与其他竞品大模型在通用领域的 **5-shot 测试**

	C-Eval	MMLU	CMMLU	Gaokao	AGIEval	BBH
	5-shot	5-shot	5-shot	5-shot	5-shot	3-shot
GPT-4	68.40	83.93	70.33	66.15	63.27	75.12
GPT-3.5 Turbo	51.10	68.54	54.06	47.07	46.13	61.59
LLaMA-13B	28.50	46.30	31.15	28.23	28.22	37.89
LLaMA2-13B	35.80	55.09	37.99	30.83	32.29	46.98
Vicuna-13B	32.80	52.00	36.28	30.11	31.55	43.04
Chinese-Alpaca-Plus-13B	38.80	43.90	33.43	34.78	35.46	28.94
XVERSE-13B	53.70	55.21	58.44	44.69	42.54	38.06
Baichuan-13B-Base	52.40	51.60	55.30	49.69	43.20	43.01
Baichuan2-13B-Base	58.10	59.17	61.97	54.33	48.17	48.78

开发：Prompt 编写

完成模型选择后，接下来就是针对每个任务**编写 Prompt。**由于我们选择的模型参数量有限，因此 Prompt 的选择和调教就显得非常重要。在结合 Prompt 编写资料和实际实验结果，我总结了以下几点原则：

- **虚拟身份**
  - 给模型设定一个具体的角色或身份，比如新闻编辑、历史学者等。这样做可以帮助模型在生成文本时保持一致的语气和风格，使输出更加自然和专业。

    - 例如，如果模型扮演新闻编辑的角色，它在总结事件时会自然地采用更加客观和精炼的语言。
- **分点论述**
    - 将复杂的事件信息结构化为几个关键点。这不仅有助于模型更清晰地理解和处理信息，还能使最终的总结更加条理清晰，便于读者理解。
    - 例如：“请按照以下格式简洁明了地总结事件：1. 时间：... 2. 地点：... 3. 主要参与者：... 4. 关键点：... 5. 事件总结：... 6. 影响及后果：... 7. 评论观点：”。
- **举例说明**
    - 在Prompt中包括具体示例可以指导模型生成更具体、更贴近实际需求的内容。这可以作为模型生成文本的模板或框架，尤其是在处理复杂或多层次的信息时。
    - 示例可以是相关领域的典型事件总结，或是以往成功的总结案例，例如：“请按照以下格式简洁明了地总结事件：1. 时间：请提供事件发生的具体时间。2. 地点：请描述事件发生的具体地点。3. 主要参与者：请列出涉及的主要人物或团体。”

后端通信：API

- **基于 Flask 的 Web 服务 API**

Flask 是一个轻量级的 Python web 框架，适合快速构建简单而强大的 web 应用。因为大模型需要更加强大且稳定的算力，因此我们选择将大模型放到GPU算力服务器上。因此，大模型所在服务器与后端不在同一台主机上。为此，我们基于 Flask 构建了一个轻量级 web 服务 API，用于后端与大模型进行通信。

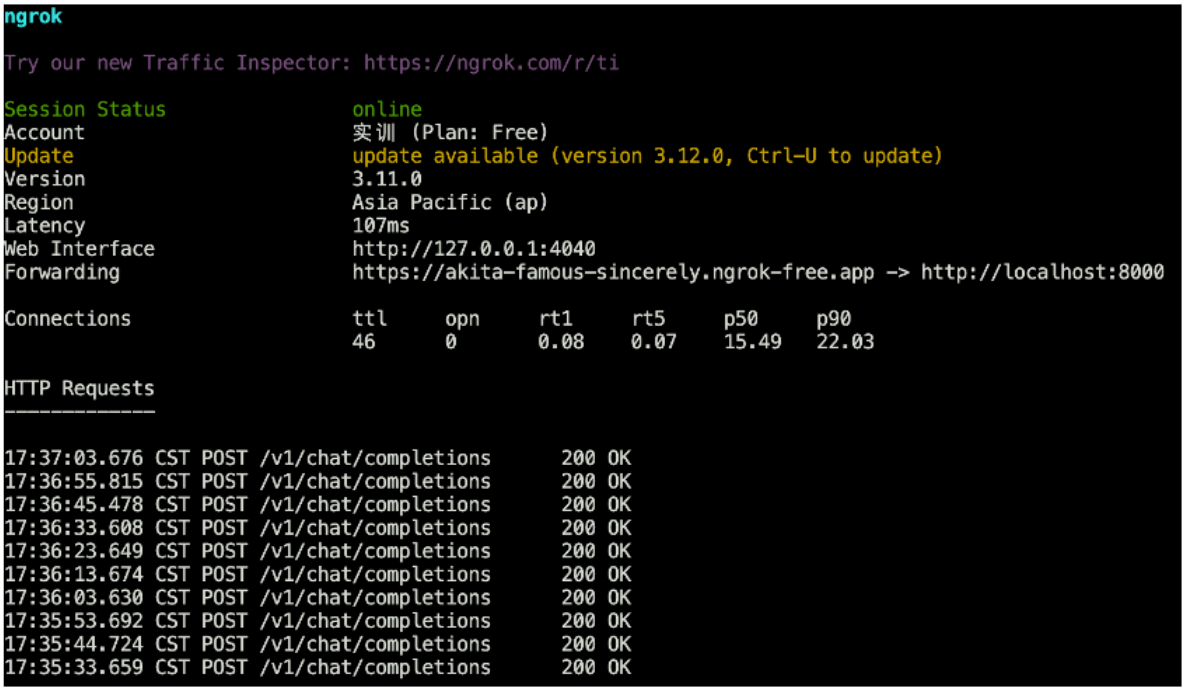
考虑到大模型通常需要较强的计算能力和稳定性，我们选择将其部署在配备 GPU 的服务器上以提高处理效率。由于大模型运行在单独的服务器上，这就要求后端必须能够有效地与之通信。为了解决这一需求，我们基于 Flask 构建了一个轻量级 web 服务 API，用于后端与大模型进行通信。Flask 是一个轻量级的 Python web 框架，非常适合快速开发简单而强大的 web 应用，因此非常符合该项目服务 API 的需求。

```
(baichuan2) root@sse21311572-2:/mnt/sse21311572/Baichuan2# python OpenAI_api.py
You are using an old version of the checkpointing format that is deprecated (We v
kwargs` in case you passed it).Please update to the new format on your modeling f
y remove the definition of the method `_set_gradient_checkpointing` in your model
 * Serving Flask app 'OpenAI_api'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:8000
 * Running on http://10.42.15.58:8000
Press CTRL+C to quit
127.0.0.1 - - [06/Jul/2024 14:04:32] "POST /v1/chat/completions HTTP/1.1" 200 -
127.0.0.1 - - [06/Jul/2024 14:04:56] "POST /v1/chat/completions HTTP/1.1" 200 -
127.0.0.1 - - [06/Jul/2024 14:05:08] "POST /v1/chat/completions HTTP/1.1" 200 -
127.0.0.1 - - [06/Jul/2024 14:05:30] "POST /v1/chat/completions HTTP/1.1" 200 -
127.0.0.1 - - [06/Jul/2024 14:05:42] "POST /v1/chat/completions HTTP/1.1" 200 -
```

- **基于 Ngrok 的内网穿透**

由于我们的服务器位于受限的学院内网中，需要通过**端口转发**才能访问。为了克服这一挑战，我们选择使用 Ngrok 工具来实现内网穿透，从而使外部系统可以安全地访问学院服务器上的服务。Ngrok 是一种强大的服务，它通过建立一个安全的隧道到 Ngrok 的公共服务器，将本地端口转发到互联网上，使得内网中的服务可以被外界访问。

我们开发了一个专用的**内网穿透脚本**，使用 Ngrok 自动设置和维护这些隧道。这个脚本负责启动 Ngrok 服务，并将其指向我们的 Flask 应用端口。通过这种方式，尽管 Flask 应用部署在内网中，外部用户和系统依然可以通过从 Ngrok 获得的公网 URL 进行访问。



叶梓荣

调研阶段

针对事件抽取任务，对不同的事件抽取模型（如DeepKE、ERNIE-UIE等）进行调研，并在本地服务器部署。对不同的模型评估其效率、效果等。

开发阶段

从零搭建项目前端部分，完成前端开发。

主要前端组件：

- Echarts：用于图表绘制
- DataV：用于非图表数据展示
- relation-graph：用于绘制事件关系图

为提高代码复用性、规范性、可读性等，将图表等可复用元素封装成组件。组件之间相互独立，提高前端程序的稳定性。